Enron Submission Free-Response Questions **Christopher P. Boone**

1. **Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: "data exploration", "outlier investigation"]**

   The ultimate goal of this project was to identify persons of interest(POI) in the dataset from the 2011 Enron Scandals. Using the database of emails and other financial data on individuals in the corporation we want to apply multiple machine learning algorithms to create a classifier that will identify any POI that may be involved in the scandal.

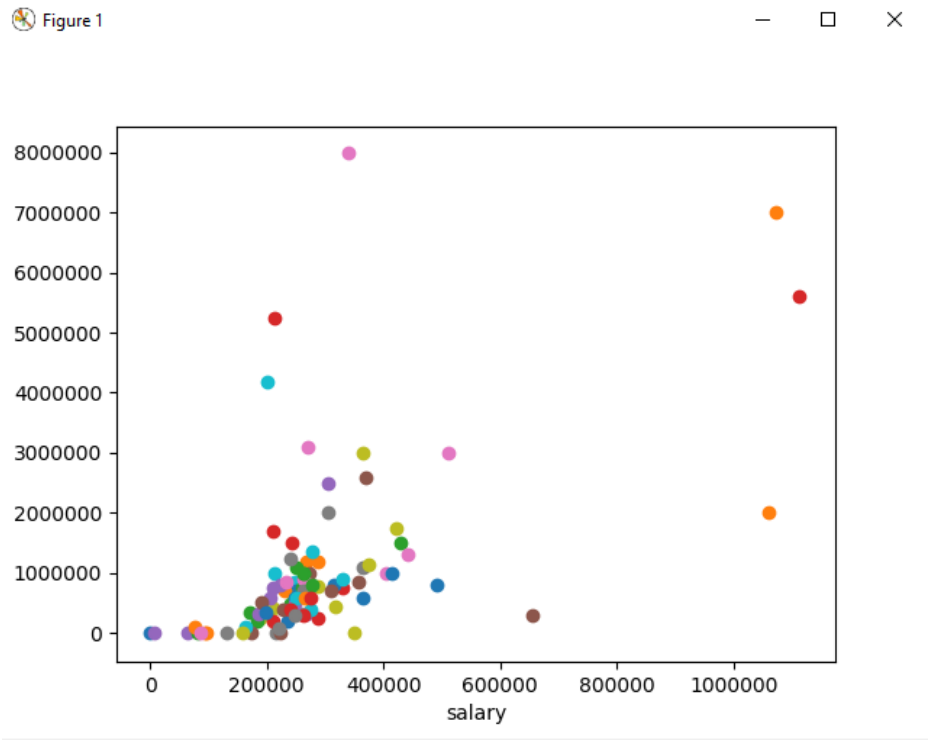
   This data set contains 21 features that have information on 146 users. The features that we are given along with ones that we created are as follows:

   Financial: salary, deferral_payments, total_payments, loan_advances, bonus, restricted_stock_deferred, deferred_income, total_stock_value, expenses, exercised_stock_options, other, long_term_incentive, restricted_stock, director_fees

   Email: to_messages, email_address, from_poi_to_this_person, from_messages, from_this_person_to_poi, from_messages

   Other Features: poi (Person of Interest Flag)

   Not all users have complete data which is represented by NaN values. To allow me to apply the algorithms and use basic mathematical functions such as mean and average I replaced the NaN values with 0. One other step I took when reviewing and cleaning up the data was to remove any outliers (The TOTAL column) to prevent any skewing of the investigation results. With the outlier removed it gave a much more accurate and clearer plot of the data that was easier to understand and see any trends with the data.



2. **What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: "create new features", "intelligently select features", "properly scale features"]**

   In my investigation I ended up creating two new features, fraction_from and fraction_to. These features allow me to see the number of emails to and from each of the user emails that are associated with a POI. The code I used is derived from some of the lessons within this course and is as follows:

```
def computeFraction( poi_messages, all_messages ):
    """ given a number messages from POI (numerator)
        and number of all messages to a person (denominator),
        return the fraction of messages to that person
        that are from/to a POI
   """

    ### number of emails from an loi / total number of emails recieved.
    fraction = 0

    if poi_messages != 'NaN' and all_messages != 'NaN':
        fraction = (poi_messages / float(all_messages))

    return fraction
```

   I called the function that I created in a 'for' loop for each employee which allowed me to build a comprehensive list of people and how much they have been communicating with know POIs.

   To determine which features to use I utilized SelectKBest where I set k=10 which in turn returns the 10 best features. The list and the respective scores that were returned when I ran this is as follows:

   Sorted scores: [('poi', inf),
   ('bonus', 15.737391429512),
   ('salary', 12.873133127161122),
   ('fraction_to', 12.04148636737257),
   ('long_term_incentive', 7.10926011090806),
   ('expenses', 3.529256419682008),
   ('from_poi_to_this_person', 3.3748319026287685),
   ('fraction_from', 1.652611956174406),
   ('from_this_person_to_poi', 1.5754557875820683),
   ('to_messages', 0.7410881696694973),
   ('from_messages', 0.3075883976668778)]

3. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]**

   The first of two algorithms that I tested was the Naïve Bayes. I chose this one due to the simplicity. Being a more simplistic algorithm to implement. It made getting comfortable with the code for working with algorithms as well as getting a firm understanding of the different ways of measuring performance a bit easier.

   NAIVE BAYES RESULTS
   accuracy_score: 0.7844736842105264
   precision: 0.5
   recall: 0.2601031746031746

   The second algorithm that I chose was the Random Forest Classifier. Of the two I tested, this one returned with the highest results of the evaluation metrics. This classifier also allows for much higher accuracy as well as will inherently handle missing values in the data helping to maintain accuracy.

   RANDOM FOREST CLASSIFIER RESULTS
   accuracy: 0.96
   precision: 1.0
   recall: 0.7142857142857143

4. **What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric items: "discuss parameter tuning", "tune the algorithm"]**

   When you say that you are tuning an algorithm you are trying to find the best set of parameters for the given algorithm that will give you the best score results and/or reducing the chance of an error in the prediction. The method that I used to tune the two algorithms that I tested was the use of Grid search. This method repeatedly builds a model using multiple combinations of parameters to determine the best combination.

5. **What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric items: "discuss validation", "validation strategy"]**

An important aspect of working with machine learning algorithms is validation. We do this to evaluate the overall performance of the algorithm to find out how well it performs and if it is a good method to approach the problem with. When validating, we want to use a test data set so that we can check the predictions against a known outcome. In this instance I used cross validation to split my data set into a smaller test group that was used to train the algorithm using the data that is leftover. One of the main classic mistakes you can make when validating is to not split your data into separate training and testing data sets. If not done right the training model may model the data too well causing overfitting.

I ended up using sklearn.cross_validations train_test_split to break the data set apart by 30% for testing. Once doing this I used the sklearn.metrics to score the performance of the algorithms to validate how well they performed.

6. **Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]**

Chosen Algorithm: Random Forest Classifier

Precision: 0.99744 **Recall**: 0.97250 **Accuracy:** 0.99538

Precision: Number of True Positives / (True Positives + False Positives)

Percentage show as decimal, out of total items marked as positive, how many of them really were positive.

Recall: Number of True Positives / (True Positives + False Negatives)

Percentage shown as decimal, out of total items that were truly positive, how many were correctly identified as positive.

Accuracy: (True Positives + True Negatives) / Total Predictions

Percentage shown as decimal, out of total number of predictions, how many were properly identified as true positives and true negatives.